NAME: Atharva Rajbanshi
NJIT UCID: ar2699
Email Address: ar2699@njit.edu
03/10/2024
Professor: Yasser Abduallah
CS 634 104 Data Mining

https://github.com/rajatharva/DataMining_midtermproject

Midterm Project Report

Implementation and Code Usage

Abstract:

Exploring the world of data mining opens up a powerful approach to uncover hidden patterns and associations within vast datasets. This project takes a unique angle by comparing three essential methods in association rule mining: Brute Force, Apriori, and FP-Tree. Let's dive into the core concepts and principles that drive this work.

The goal of each method is to create associations within the data. I started by identifying the most frequent items within our list of transactions. Then, based on the user's support parameter, calculated the support for each item, removing those that didn't meet the specified threshold.

The Brute Force method takes a simple, yet exhaustive approach by checking all possible combinations of items to find frequent itemsets and generate association rules. While straightforward, this method can be time-consuming, especially for large datasets.

The Apriori Algorithm gradually finds frequent itemsets, increasing the size of itemsets iteratively and filtering out those that don't meet a minimum support threshold. It strikes a balance between efficiency and effectiveness in discovering associations.

FP-Tree (FP-Growth) presents another popular algorithm, using a tree structure to encode the dataset and mine frequent itemsets efficiently. By creating a condensed representation of the dataset, FP-Tree reduces the need for multiple database scans, making it particularly effective for large datasets.

In this project, I applied all three methods to a custom dataset linked with a retail store. This enabled us to compare their performance in finding frequent itemsets and association rules. Key steps in this process included:

- Initializing dictionaries for candidate and frequent itemsets.

- Loading the dataset and itemsets from CSV files.

- Preprocessing the dataset to ensure item order and uniqueness.

- Collecting user input for minimum support and confidence thresholds.

- Implementing each method to generate frequent itemsets and association rules.

Through this comparative analysis, we've gained valuable insights into the strengths and weaknesses of each method. This provides a nuanced understanding of their applicability in uncovering valuable associations within retail transactions.

Core Concepts and Principles:

Discovery of Frequent Itemsets:

At the core of my project lies the exploration of three fundamental methods in association rule mining: Brute Force, Apriori, and FP-Tree. Each method aims to unveil frequent itemsets, which are sets of items that commonly co-occur in transactions. These itemsets provide crucial insights into customer purchase behavior and preferences.

Understanding Support and Confidence:

Support and confidence are key metrics in data mining, guiding my analysis and decision-making. Support measures the frequency of occurrence for an item or itemset, while confidence assesses the likelihood of items being purchased together.

Unveiling Association Rules:

The identification of strong association rules is pivotal for understanding item associations and optimizing sales strategies. These rules highlight which items are frequently purchased together, enabling targeted marketing and personalized recommendations.

Project Workflow:

My project follows a structured workflow, encompassing the implementation of Brute Force, Apriori, and FP-Tree methods:

Data Loading and Preprocessing:

I begin by loading transaction data from retail store datasets, with each transaction containing a list of items purchased by a customer. To ensure data integrity, I preprocess the datasets by filtering unique items and arranging them in a predefined order.

Setting Minimum Support and Confidence Levels:

User input plays a crucial role in my data mining endeavor. I gather user preferences for minimum support and confidence levels, essential for filtering out less significant patterns.

Iterative Generation of Candidate Itemsets:

The iterative application of the Brute Force, Apriori, and FP-Tree methods involves generating candidate itemsets of varying sizes. I commence with single items (itemset size K = 1) and progress to K = 2, K = 3, and so forth. This iterative process employs a 'brute force' method to generate all possible combinations of itemsets.

Calculation of Support Counts:

For each candidate itemset, I compute its support by tallying the number of transactions containing the itemset. Itemsets that meet the minimum support threshold are retained, while others are discarded.

Evaluation of Confidence:

I assess the confidence of association rules, indicating the strength of relationships between items. This step requires a meticulous comparison of support values for individual items and itemsets.

Extraction of Association Rules:

Association rules meeting both the minimum support and minimum confidence criteria are extracted. These rules provide invaluable insights into frequently associated items.

Results and Evaluation:

The effectiveness and efficiency of my project are evaluated based on performance metrics such as support, confidence, and the resultant association rules. I also conduct a comparison between custom implementations of Brute Force, Apriori, and FP-Tree with their respective libraries to ascertain their reliability.

Conclusion:

In conclusion, this project showcases the practical application of data mining concepts, principles, and methodologies. I have successfully employed the Brute Force, Apriori, and FP-Tree methods to derive meaningful association rules from retail transaction data. The iterative, 'brute force' approach, along with custom algorithm designs and adherence to user-defined parameters, underscores the potency of data mining in uncovering valuable patterns for informed decision-making within the retail sector.

Here are what the csv files (This program takes in 5 separate csv files: Item Names & Transactions).

Figure 1 : Amazon CSV file.

amazon

Transaction ID	Books
Trans1	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans2	A Beginner's Guide, Java: The Complete Reference, Java For Dummies
Trans3	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans4	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition, Beginning Programming with Java
Trans5	Android Programming: The Big Nerd Ranch, Beginning Programming with Java, Java 8 Pocket Guide
Trans6	A Beginner's Guide, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans7	A Beginner's Guide, Head First Java 2nd Edition, Beginning Programming with Java
Trans8	Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans9	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition, Beginning Programming with Java
Trans10	Beginning Programming with Java, Java 8 Pocket Guide, C++ Programming in Easy Steps
Trans11	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans12	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, HTML and CSS: Design and Build Websites
Trans13	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Java 8 Pocket Guide, HTML and CSS: Design and Build Websites
Trans14	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans15	Java For Dummies, Android Programming: The Big Nerd Ranch
Trans16	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans17	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans18	Head First Java 2nd Edition, Beginning Programming with Java, Java 8 Pocket Guide
Trans19	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans20	A Beginner's Guide, Java: The Complete Reference, Java For Dummies

Figure 2 : BestBuy CSV file.

BestBuy

Transaction ID	Items
Trans1	Desk Top, Printer, Flash Drive, Microsoft Office, Speakers, Anti-Virus
Trans2	Lab Top, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus
Trans3	Lab Top, Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive
Trans4	Lab Top, Printer, Flash Drive, Anti-Virus, External Hard-Drive, Lab Top Case
Trans5	Lab Top, Flash Drive, Lab Top Case, Anti-Virus
Trans6	Lab Top, Printer, Flash Drive, Microsoft Office
Trans7	Desk Top, Printer, Flash Drive, Microsoft Office
Trans8	Lab Top, External Hard-Drive, Anti-Virus
Trans9	Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, Speakers, External Hard-Drive
Trans10	Digital Camera, Lab Top, Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, External Hard-Drive, Speakers
Trans11	Lab Top, Desk Top, Lab Top Case, External Hard-Drive, Speakers, Anti-Virus
Trans12	Digital Camera, Lab Top, Lab Top Case, External Hard-Drive, Anti-Virus, Speakers
Trans13	Digital Camera, Speakers
Trans14	Digital Camera, Desk Top, Printer, Flash Drive, Microsoft Office
Trans15	Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, Speakers, External Hard-Drive
Trans16	Digital Camera, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive, Speakers
Trans17	Digital Camera, Lab Top, Lab Top Case
Trans18	Digital Camera, Lab Top Case, Speakers
Trans19	Digital Camera, Lab Top, Printer, Flash Drive, Microsoft Office, Speakers, Lab Top Case, Anti-Virus
Trans20	Digital Camera, Lab Top, Speakers, Anti-Virus, Lab Top Case

Figure 3 : kmart CSV file.

Kmart

Transaction ID	Items
Trans1	Decorative Pillows, Quilts, Embroidered Bedspread
Trans2	Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections, Bed Skirts, Bedspreads, Sheets
Trans3	Decorative Pillows, Quilts, Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections
Trans4	Kids Bedding, Bedding Collections, Sheets, Bedspreads, Bed Skirts
Trans5	Decorative Pillows, Kids Bedding, Bedding Collections, Sheets, Bed Skirts, Bedspreads
Trans6	Bedding Collections, Bedspreads, Bed Skirts, Sheets, Shams, Kids Bedding
Trans7	Decorative Pillows, Quilts
Trans8	Decorative Pillows, Quilts, Embroidered Bedspread
Trans9	Bedspreads, Bed Skirts, Shams, Kids Bedding, Sheets
Trans10	Quilts, Embroidered Bedspread, Bedding Collections
Trans11	Bedding Collections, Bedspreads, Bed Skirts, Kids Bedding, Shams, Sheets
Trans12	Decorative Pillows, Quilts
Trans13	Embroidered Bedspread, Shams
Trans14	Sheets, Shams, Bed Skirts, Kids Bedding
Trans15	Decorative Pillows, Quilts
Trans16	Decorative Pillows, Kids Bedding, Bed Skirts, Shams
Trans17	Decorative Pillows, Shams, Bed Skirts
Trans18	Quilts, Sheets, Kids Bedding
Trans19	Shams, Bed Skirts, Kids Bedding, Sheets
Trans20	Decorative Pillows, Bedspreads, Shams, Sheets, Bed Skirts, Kids Bedding

Figure 4 : Nike CSV file.

nike

Transaction ID	Items
Trans1	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans2	Running Shoe, Socks, Sweatshirts
Trans3	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans4	Running Shoe, Sweatshirts, Modern Pants
Trans5	Running Shoe, Socks, Sweatshirts, Modern Pants, Soccer Shoe
Trans6	Running Shoe, Socks, Sweatshirts
Trans7	Running Shoe, Socks, Sweatshirts, Modern Pants, Tech Pants, Rash Guard, Hoodies
Trans8	Swimming Shirt, Socks, Sweatshirts
Trans9	Swimming Shirt, Rash Guard, Dry Fit V-Nick, Hoodies, Tech Pants
Trans10	Swimming Shirt, Rash Guard, Dry
Trans11	Swimming Shirt, Rash Guard, Dry Fit V-Nick
Trans12	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Hoodies, Tech Pants, Dry Fit V-Nick
Trans13	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Tech Pants, Dry Fit V-Nick, Hoodies
Trans14	Running Shoe, Swimming Shirt, Rash Guard, Tech Pants, Hoodies, Dry Fit V-Nick
Trans15	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Dry Fit V-Nick, Rash Guard, Tech Pants
Trans16	Swimming Shirt, Soccer Shoe, Hoodies, Dry Fit V-Nick, Tech Pants, Rash Guard
Trans17	Running Shoe, Socks
Trans18	Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Rash Guard, Tech Pants, Dry Fit V-Nick
Trans19	Running Shoe, Swimming Shirt, Rash Guard
Trans20	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Tech Pants, Rash Guard, Dry Fit V-Nick

Figure 5 : Generic CSV file.

Generic

Transaction ID	Items
Trans1	A, B, C
Trans2	A, B, C
Trans3	A, B, C, D
Trans4	A, B, C, D, E
Trans5	A, B, D, E
Trans6	A, D, E
Trans7	A, E
Trans8	A, E
Trans9	A, C, E
Trans10	A, C, E
Trans11	A, C, E

Below are screenshots of the code from python file:

Importing Necessary libraries

```
import csv
import itertools
import time
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
import pandas as pd
```

Brute Force Method:

```
#Brute Force
def brute_force(transactions, min_support, min_confidence):
    items = set(item for transaction in transactions for item in transaction)
    itemsets = []
    for i in range(1, len(items) + 1):
        itemsets.extend(itertools.combinations(items, i))
    frequent_itemsets = {}
    for itemset in itemsets:
        frequency = sum(1 for transaction in transactions if set(itemset).issubset(transaction))
        if frequency / len(transactions) >= min_support:
            frequent_itemsets[itemset] = frequency
    return generate_association_rules(frequent_itemsets, transactions, min_confidence)
```

Defining Association rule generation function:

```
def generate_association_rules(frequent_itemsets, transactions, min_confidence):
    rules = []
    for itemset, frequency in frequent_itemsets.items():
        for i in range(1, len(itemset)):
            for antecedent in itertools.combinations(itemset, i):
                consequent = set(itemset) - set(antecedent)
                antecedent_transactions = sum(1 for transaction in transactions if set(antecedent).issubset(transact
                if antecedent_transactions > 0:
                    confidence = frequency / antecedent_transactions
                    if confidence >= min_confidence:
                        rules.append((antecedent, consequent, confidence))
    return rules
```

Reading csv files and printing association rule function

```python
def read_data(filename):
    transactions = []
    with open(filename, 'r') as file:
        csv_reader = csv.reader(file)
        next(csv_reader)  # Skip header row
        for row in csv_reader:
            transactions.append(row[1].split(", "))
    return transactions

def print_rules(rules, method):
    print(f"{method} Association Rules:")
    for rule in rules:
        antecedents = ', '.join(rule[0])
        consequents = ', '.join(rule[1])
        print(f"{antecedents} -> {consequents}, Confidence: {rule[2]:.2f}")
    print("\n")
```

Apriori and FP-Tree Method:

```python
def run_apriori_fpgrowth(transactions, min_support, min_confidence):
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)

    #Apriori

    start_time = time.time()
    frequent_itemsets_apriori = apriori(df, min_support=min_support, use_colnames=True)
    rules_apriori = association_rules(frequent_itemsets_apriori, metric="confidence", min_threshold=min_confidence)
    end_time = time.time()
    print(f"Apriori Execution Time: {end_time - start_time} seconds")
    if not rules_apriori.empty:
        for index, row in rules_apriori.iterrows():
            print(f"{' , '.join(row['antecedents'])} -> {' , '.join(row['consequents'])}, Confidence: {row['confiden
    else:
        print("No association rules found.")
    print("\n")

    # FP-Growth

    start_time = time.time()
    frequent_itemsets_fp = fpgrowth(df, min_support=min_support, use_colnames=True)
    rules_fp = association_rules(frequent_itemsets_fp, metric="confidence", min_threshold=min_confidence)
    end_time = time.time()
    print(f"FP-Growth Execution Time: {end_time - start_time} seconds")
    if not rules_fp.empty:
        for index, row in rules_fp.iterrows():
            print(f"{' , '.join(row['antecedents'])} -> {' , '.join(row['consequents'])}, Confidence: {row['confiden
    else:
        print("No association rules found.")
    print("\n")
```

Main:

```python
def main():
    datasets = ["amazon.csv", "BestBuy.csv", "Kmart.csv", "nike.csv", "Generic.csv"]
    dataset_choice = int(input("Choose a dataset (1-5): \n1.Amazon\n2.BestBuy\n3.Kmart\n4.Nike\n5.Generic\n"))
    min_support = float(input("Enter minimum support (as a decimal): "))
    min_confidence = float(input("Enter minimum confidence (as a decimal): "))

    transactions = read_data(datasets[dataset_choice - 1])

    # Brute Force
    start_time = time.time()
    rules_brute_force = brute_force(transactions, min_support, min_confidence)
    end_time = time.time()
    print(f"\nBrute Force Execution Time: {end_time - start_time} seconds")
    print_rules(rules_brute_force, "Brute Force")

    # Apriori and FP-Growth
    run_apriori_fpgrowth(transactions, min_support, min_confidence)

if __name__ == "__main__":
    main()
```

Below are screenshots to show that the program runs in the Terminal:

For Amazon Transactions:

```
Choose a dataset (1-5):
1.Amazon
2.BestBuy
3.Kmart
4.Nike
5.Generic
1
Enter minimum support (as a decimal): 0.5
Enter minimum confidence (as a decimal): 0.5

Brute Force Execution Time: 0.0071620941162109375 seconds
Brute Force Association Rules:
Java For Dummies -> Java: The Complete Reference, Confidence: 0.77
Java: The Complete Reference -> Java For Dummies, Confidence: 1.00


Apriori Execution Time: 0.006915092468261719 seconds
Java For Dummies -> Java: The Complete Reference, Confidence: 0.77
Java: The Complete Reference -> Java For Dummies, Confidence: 1.00


FP-Growth Execution Time: 0.008121967315673828 seconds
Java For Dummies -> Java: The Complete Reference, Confidence: 0.77
Java: The Complete Reference -> Java For Dummies, Confidence: 1.00
```

For BestBuy transactions:

```
Brute Force Execution Time: 0.011825799942016602 seconds
Brute Force Association Rules:
Flash Drive -> Printer, Confidence: 0.77
Printer -> Flash Drive, Confidence: 1.00
Flash Drive -> Microsoft Office, Confidence: 0.85
Microsoft Office -> Flash Drive, Confidence: 1.00
Flash Drive -> Anti-Virus, Confidence: 0.77
Anti-Virus -> Flash Drive, Confidence: 0.71
Lab Top -> Lab Top Case, Confidence: 0.83
Lab Top Case -> Lab Top, Confidence: 0.71
Lab Top -> Anti-Virus, Confidence: 0.83
Anti-Virus -> Lab Top, Confidence: 0.71
Lab Top Case -> Anti-Virus, Confidence: 0.86
Anti-Virus -> Lab Top Case, Confidence: 0.86
```

```
Apriori Execution Time: 0.007980108261108398 seconds
Anti-Virus -> Flash Drive, Confidence: 0.71
Flash Drive -> Anti-Virus, Confidence: 0.77
Anti-Virus -> Lab Top, Confidence: 0.71
Lab Top -> Anti-Virus, Confidence: 0.83
Anti-Virus -> Lab Top Case, Confidence: 0.86
Lab Top Case -> Anti-Virus, Confidence: 0.86
Flash Drive -> Microsoft Office, Confidence: 0.85
Microsoft Office -> Flash Drive, Confidence: 1.00
Flash Drive -> Printer, Confidence: 0.77
Printer -> Flash Drive, Confidence: 1.00
Lab Top -> Lab Top Case, Confidence: 0.83
Lab Top Case -> Lab Top, Confidence: 0.71
```

```
FP-Growth Execution Time: 0.005619049072265625 seconds
Anti-Virus -> Lab Top Case, Confidence: 0.86
Lab Top Case -> Anti-Virus, Confidence: 0.86
Anti-Virus -> Flash Drive, Confidence: 0.71
Flash Drive -> Anti-Virus, Confidence: 0.77
Flash Drive -> Microsoft Office, Confidence: 0.85
Microsoft Office -> Flash Drive, Confidence: 1.00
Flash Drive -> Printer, Confidence: 0.77
Printer -> Flash Drive, Confidence: 1.00
Anti-Virus -> Lab Top, Confidence: 0.71
Lab Top -> Anti-Virus, Confidence: 0.83
Lab Top -> Lab Top Case, Confidence: 0.83
Lab Top Case -> Lab Top, Confidence: 0.71
```

For Kmart Transactions:

```
Brute Force Execution Time: 0.0065460205078125 seconds
Brute Force Association Rules:
Bed Skirts -> Kids Bedding, Confidence: 0.91
Kids Bedding -> Bed Skirts, Confidence: 0.83
Sheets -> Kids Bedding, Confidence: 1.00
Kids Bedding -> Sheets, Confidence: 0.83
```

```
Apriori Execution Time: 0.006983041763305664 seconds
Bed Skirts -> Kids Bedding, Confidence: 0.91
Kids Bedding -> Bed Skirts, Confidence: 0.83
Kids Bedding -> Sheets, Confidence: 0.83
Sheets -> Kids Bedding, Confidence: 1.00
```

```
FP-Growth Execution Time: 0.004857063293457031 seconds
Bed Skirts -> Kids Bedding, Confidence: 0.91
Kids Bedding -> Bed Skirts, Confidence: 0.83
Kids Bedding -> Sheets, Confidence: 0.83
Sheets -> Kids Bedding, Confidence: 1.00
```

For Nike Transactions:

```
Brute Force Execution Time: 0.022221088409423828 seconds
Brute Force Association Rules:
Modern Pants -> Sweatshirts, Confidence: 1.00
Sweatshirts -> Modern Pants, Confidence: 0.77
Rash Guard -> Swimming Shirt, Confidence: 0.83
Swimming Shirt -> Rash Guard, Confidence: 0.91
Sweatshirts -> Socks, Confidence: 0.92
Socks -> Sweatshirts, Confidence: 0.92
Sweatshirts -> Running Shoe, Confidence: 0.85
Running Shoe -> Sweatshirts, Confidence: 0.79
Socks -> Running Shoe, Confidence: 0.85
Running Shoe -> Socks, Confidence: 0.79
Sweatshirts -> Socks, Running Shoe, Confidence: 0.77
Socks -> Running Shoe, Sweatshirts, Confidence: 0.77
Running Shoe -> Socks, Sweatshirts, Confidence: 0.71
Sweatshirts, Socks -> Running Shoe, Confidence: 0.83
Sweatshirts, Running Shoe -> Socks, Confidence: 0.91
Socks, Running Shoe -> Sweatshirts, Confidence: 0.91
```

```
Apriori Execution Time: 0.008249998092651367 seconds
Modern Pants -> Sweatshirts, Confidence: 1.00
Sweatshirts -> Modern Pants, Confidence: 0.77
Swimming Shirt -> Rash Guard, Confidence: 0.91
Rash Guard -> Swimming Shirt, Confidence: 0.83
Socks -> Running Shoe, Confidence: 0.85
Running Shoe -> Socks, Confidence: 0.79
Running Shoe -> Sweatshirts, Confidence: 0.79
Sweatshirts -> Running Shoe, Confidence: 0.85
Socks -> Sweatshirts, Confidence: 0.92
Sweatshirts -> Socks, Confidence: 0.92
Socks , Running Shoe -> Sweatshirts, Confidence: 0.91
Socks , Sweatshirts -> Running Shoe, Confidence: 0.83
Running Shoe , Sweatshirts -> Socks, Confidence: 0.91
Socks -> Running Shoe , Sweatshirts, Confidence: 0.77
Running Shoe -> Socks , Sweatshirts, Confidence: 0.71
Sweatshirts -> Socks , Running Shoe, Confidence: 0.77
```

For Generic Transactions:

```
Brute Force Execution Time: 0.0003609657287597656 seconds
Brute Force Association Rules:
A -> C, Confidence: 0.64
C -> A, Confidence: 1.00
A -> E, Confidence: 0.73
E -> A, Confidence: 1.00

Apriori Execution Time: 0.01024007797241211 seconds
C -> A, Confidence: 1.00
A -> C, Confidence: 0.64
A -> E, Confidence: 0.73
E -> A, Confidence: 1.00

FP-Growth Execution Time: 0.005077362060546875 seconds
C -> A, Confidence: 1.00
A -> C, Confidence: 0.64
A -> E, Confidence: 0.73
E -> A, Confidence: 1.00
```

Other

The source code (.py file) and data sets (.csv files) will be attached to the zip file. *Link to Git Repository*

https://github.com/rajatharva/DataMining_midtermproject